1. Desarrolle un programa que permita obtener estadística básica de una lista de notas ingresadas por un usuario. Para esto considere que:

a. Las notas que ingresa el usuario en la pantalla deben estar entre 1.0 y 7.0, ambos incluidos.
b. En el caso que se ingrese una nota incorrecta, se le debe indicar al usuario del error y solicitar que ingrese una nueva nota.
c. El usuario puede ingresar la cantidad de notas que quiera. Asuma que para indicar que ya ha terminado de ingresar las notas, el usuario ingresa el valor 0.
d. Una vez ingresadas todas las notas se debe generar un informe que muestre la frecuencia y porcentaje para rangos de notas de al menos 10 décimas. Un ejemplo de ejecución del algoritmo sería el siguiente:

```
Ingrese las notas entre 1.0 y 7.0:
1.9
5.5
7.0
Nota ingresada fuera de rango, vuelva a ingresar.
6.9
4.0
5.0
5.9
3.8
3.2
1.5
0
Total de notas ingresadas: 10
Detalle:
[1.0-1.9]: 2 | 20.0 %
[2.0-2.9]: 0 | 00.0 %
[3.0-3.9]: 2 | 20.0 %
[4.0-4.9]: 1 | 20.0 %
[5.0-5.9]: 3 | 30.0 %
[6.0-7.0]: 2 | 20.0 %
```

Sugerencias:
Desarrolle su programa utilizando los siguientes elementos:

1. Listas.
2. Funciones recursivas.

2. Desarrolle una calculadora que permita realizar las siguientes operaciones entre dos números: (1) suma, (2) resta, (3) multiplicación, (4) división, y (5) promedio. El programa a desarrollar debe cumplir los siguientes requisitos:

a. Preguntar al usuario los dos números sobre los que se realizará una operación matemática.
b. Preguntar al usuario qué operación matemática desea hacer.
c. Cada vez que el usuario ya seleccionó la operación, se debe mostrar el resultado y preguntar si desea realizar otra operación o bien, ingresar otro par de números para realizar alguna de las operaciones.
d. Además de realizar operaciones o ingresar otros números, se debe dar la opción al usuario de terminar con el uso de la calculadora. Una vez que sale se debe dar un mensaje de despedida.

Con la información anterior, desarrolle un programa dé respuesta bajo la siguiente estructura:

- pregunta_1

- main.py
- operaciones.py

Sugerencias:
Desarrolle su programa utilizando los siguientes elementos:

1. Módulos
2. Una función por cada operación.

3. Desarrolle un programa en el que se calcule la diferencia entre dos fechas ingresadas. Para su desarrollo considere los siguientes requisitos que debe cumplir el programa:

a. El programa debe permitir ingresar la fecha **bajo uno de los siguientes dos formatos**: (1) se ingresan el día, mes y año de manera separada o, (2) se ingresan el día, mes y año una sola vez según la estructura ddmmaaaa (por ejemplo, para la fecha 19 de Junio del 2019 se ingresa 19062019).
b. El usuario puede decidir mostrar la diferencia entre las dos fechas en términos de meses y años.
c. Por simplicidad: (1) asuma que, si al usuario se le pide ingresar la fecha inicial y fecha final, la primera ingresada es menor a la segunda (note que en ningún caso será iguales), (2) no existen años bisiestos, y (3) no hay error de tipeo en el ingreso de fechas.

Con la información anterior, desarrolle un programa que dé respuesta a lo solicitado bajo la siguiente estructura:

- pregunta_2
  - main.py
  - fechas.py

Sugerencias:
Desarrolle su programa utilizando los siguientes elementos:

1. Módulos
2. Analice bien el cálculo de la diferencia entre fechas. Tenga especial cuidado con los meses y días.
3. Se sugiere preguntar por separado el día, el mes y el año de las fechas.

4. Desarrolle un programa de *Login* en donde se le pide al usuario un correo electrónico y la contraseña. El programa deberá ser capaz de:

a. Verificar si el correo electrónico está registrado en el sistema. En el caso que no exista, deberá pedir el ingreso de uno correcto, mientras que, para el caso contrario, deberá comprobar su correspondencia con la contraseña según lo indicado en la **parte b**.
b. Verificar que la contraseña corresponde al correo electrónico ingresado. Si corresponde, deberá entregar un mensaje que la contraseña es correcta y que ha ingresado al sistema. En caso contrario, deberá mostrar un mensaje indicando que la contraseña es incorrecta y que debe ingresarla nuevamente (esto debe hacerlo hasta que se ingrese la contraseña correcta).
c. Una vez ingresado al sistema, desplegar un menú donde el usuario pueda seleccionar si modificar la contraseña o el nombre de usuario (asuma que todos los correos serán @correo.cl). El programa debe ser capaz de captar cuál es la opción seleccionada por el usuario. También debe dar la opción de salir del sistema.
d. Verificar que el nuevo nombre de usuario del correo o la contraseña deban ser exactamente de largo 8. Si no se cumple este requisito, se deberá solicitar nuevamente el ingreso, además de indicar que estos deben ser del largo especificado anteriormente.
e. Guardar en login.txt las modificaciones que se realizaron a los correos electrónicos y contraseñas.

Para el desarrollo del programa, considere que usted cuenta con el archivo login.txt, en el que se enlista los correos electrónicos y sus respectivas contraseñas. En archivo, **cada línea** representa a un usuario registrado bajo la siguiente estructura: <correo>;<contraseña>. A continuación, se muestra un ejemplo de los 3 primeros usuarios registrados:

```
ionceran@correo.cl;o5jmq88a
matinget@correo.cl;6to4cg1j
mimengue@correo.cl;85rbkqn8
```

Note que todos los correos se componen de un nombre de 8 caracteres más el @correo.cl, las contraseñas son alfanuméricas y de largo 8, y que el correo electrónico y la contraseña se separan por el carácter ; (punto y coma). Además, recuerde que los saltos de líneas se importan al final como el conjunto de caracteres "\n".

Sugerencias:
Desarrolle su programa utilizando los siguientes elementos:

1. Manejo de archivos. Se recomienda manejar: open(), readlines(), write(), writelines(), close().
2. Slicing. En esta parte, tenga cuidado con el salto de línea que se importará cuando se importe la contraseña. Puede utilizar el método index para identificarlo, o bien, como ya se sabe que la contraseña será de 8, por medio de slicing, solo extraer 8 caracteres desde el ;.
3. Note que cada letra de la pregunta es una característica solicitada del sistema. Se recomienda que identifique las que más maneja y trabajar sobre ellas. Si no funciona al correr el programa, se recomienda dejar clara la lógica que se desea implementar, puesto que la asignación del puntaje es gradual, no dicotómica.

5. Desarrolle un programa que permita generar un reporte de estadística de las notas que se reciben en un txt. Imagine que la estructura del txt es la siguiente:

```
Christiane Endler;6.5
Juan Perez;5.0
Alexis Sanchez;5.5
…
```

Se pide que el programa lea este archivo y entregue un reporte que muestre la siguiente información:

a. Total y porcentaje de aprobados y reprobados.
b. Total y porcentaje de notas entre los rangos: 1.0-1.9, 2.0-2.9, 3.0-3.9, 4.0-4.9, 5.0-5.9, 6.0-7.0
c. Lista de alumnos en el cuartil 1, cuartil 2, cuartil 3 y cuartil 4.

Sugerencias:
Desarrolle su programa utilizando los siguientes elementos:

1. Manejo de archivos. Se recomienda manejar: open(), readlines(), write(), writelines(), close().
2. Slicing. En esta parte, tenga cuidado con el salto de línea que se importará cuando se importe la nota. Puede utilizar el método index para identificarlo, o bien, como ya se sabe que la nota siempre será de 3 caracteres, puede utilizar distintos métodos para importar bien. Recuerde que el readlines() entrega los elementos como string, pero la nota se debe trabajar como float.
3. Note que cada letra de la pregunta es una característica solicitada del sistema. Se recomienda que identifique las que más maneja y trabajar sobre ellas. Si no funciona al correr el programa, se recomienda dejar clara la lógica que se desea implementar, puesto que la asignación del puntaje es gradual, no dicotómica.

6. Desarrolle un programa que permita a un usuario registrarse o iniciar sesión para jugar con el computador al cachipún. La idea es que el usuario pueda escoger entre tres opciones (piedra, papel o tijera) y que el computador, de forma aleatoria, también haga su elección. Posteriormente se debe mostrar en pantalla la elección de cada uno y el resultado (Gana Jugador, Gana Computador o Empate). Recuerde que Piedra gana a Tijera, Tijera gana a Papel y Papel gana a Piedra. El programa debe permitir jugar al usuario la cantidad de veces que desee. Una vez que el usuario termine de realizar todas las jugadas que desee, el programa debe realizar lo siguiente:

a. Guardar o actualizar el número de veces que el jugador ha jugado con el programa y la cantidad de veces que ha ganado.
b. Mostrar una lista de todos los jugadores que han usado el programa, mostrando primero aquel jugador que ha ganado más veces en términos relativos (total ganados dividido por total jugados). En la lista debe aparecer el nombre del jugador, la cantidad de veces jugadas, cantidad de veces ganadas y desempeño (total ganado/total jugado).

Sugerencias:
Desarrolle su programa utilizando los siguientes elementos:

1. Manejo de archivos. Se recomienda manejar: open(), readlines(), write(), writelines(), close().
2. Slicing. En esta parte, tenga cuidado con el salto de línea que se importará cuando se importe la nota. Puede utilizar el método index para identificarlo, o bien, como ya se sabe que la nota siempre será de 3 caracteres, puede utilizar distintos métodos para importar bien. Recuerde que el readlines() entrega los elementos como string, pero la nota se debe trabajar como float.
3. Note que cada letra de la pregunta es una característica solicitada del sistema. Se recomienda que identifique las que más maneja y trabajar sobre ellas. Si no funciona al correr el programa, se recomienda dejar clara la lógica que se desea implementar, puesto que la asignación del puntaje es gradual, no dicotómica.
4. Note que no se está pidiendo guardar en orden, si no que mostrar de forma ordenada.